INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

# Prototype P6.16

Per Öberg, Lars Eriksson (LiU)
November, 2012

●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●

# Free Modelica Libraries resulting from the combustion engine use case

## 1. Summary

The objectives and results of this task and deliverable are twofold:

- Evaluate how the increased flexibility that the Modelica language gives can be used when applied on a previously available Matlab implementation.
- Drive the development of the Modelica tools by presenting a challenging use case which takes advantage of advanced modeling language features. For this case that means the features of the Media and Fluid libraries.

The implementation uses models from a previously available Matlab implementation but adds the use of possible multi-component gas models instead of ideal gases with fixed gas properties. Because of the exchangeable gas models the same models can be used at many stages in the engineering process by reducing or increasing the model complexity as appropriate. The developed library depends heavily on the MSL Media Library which gives a good test case for Modelica tools. More specifically the **`replaceable package`** feature is extensively used.

## 2. MVEM Lib

The basic classes for the Mean Value Engine Library, MVEM Lib are

- GasPort
- TwoPort
- FixedVolume
- IdealRestriction (and NonIdealRestriction)
- FuelAirMixture
- Consistant set of models for
- Fuel
- Air
- Unburned gas
- Burned gas
- FuelAirMixer
- AdiabaticBurner

Using these classes combustion engine components can be created, such as

- Compressible restrictions such as, butterfly throttle, turbine waste gate, and valves.
- Turbocharger, consisting of compressor, shaft, and turbine.
- Intermediate volumes such as intake and exhaust manifold
- Pipes with heat transfer such as exhaust system and intercooler
- Forced mass flows such as from speed-density air flow calculation or fuel injection
- Torque production such as from the otto engine efficiency calculation

### 2.1. Library details

Here a short walkthrough of the library is given. For a deeper view the interested reader is directed towards the actual library code.

#### 2.1.1. GasPort

The GasPort is similar to the FluidPort model of MSL and is used as base class for 0D modeling of fluid flow

```
connector GasPort
  replaceable package Medium =
      Modelica.Media.Interfaces.PartialMedium;
  SI.Pressure p "Pressure in the connector point";
  SI.Temperature T "Temperature in the connector point";
  SI.MassFraction Xi[Medium.nXi] "Mass fractions …";
  flow SI.EnthalpyFlowRate dH "Specific enthalpy flow through the connector";
  flow SI.MassFlowRate dm "Total Mass flow through the connector";
  flow SI.MassFlowRate dmXi[Medium.nXi] "Mass flows … ";
end GasPort;
```

### 2.1.2. TwoPort

The TwoPort model is used as base class for flow components such as restrictions and sensors, much like the electrical two-port.

```
partial model TwoPort
  replaceable package Medium =
      Modelica.Media.Interfaces.PartialMedium
  Interfaces.GasPort InPut(redeclare replaceable package Medium = Medium)
  Interfaces.GasPort OutPut(redeclare replaceable package Medium = Medium)
end TwoPort;
```

### 2.1.3. FixedVolume

The FixedVolume model represents a vessel with fixed size that keeps track of energy and masses by simple bookkeeping.

```
class FixedVolume extends Basic.Restrictions.Partial.TwoPort;
  Medium.BaseProperties gas
protected
  SI.Mass m "Mass of system";
  SI.Volume V "Volume of system";
  SI.Mass mXi[Medium.nXi] "Mass of respective independent gas component";
  SI.InternalEnergy U;
equation
  …
  mXi = gas.Xi * m;
  der(m)  = InPut.dm  + OutPut.dm;
  der(mXi) = InPut.dmXi + OutPut.dmXi;
  der(U) = InPut.dH + OutPut.dH;
  U = m*gas.u;
  V = VStart;
  gas.p * V = m * gas.R * gas.T;
end FixedVolume;
```

### 2.1.4. IdealRestriction

The IdealRestriction is used as base class for compressible and incompressible flow and is ideal in the sense that enthalpy flows right through, i.e. there is no energy that dissipates from the restriction.

```
partial class IdealRestriction "Partial model for all restrictions."
   extends MVEMLib.Basic.Restrictions.Partial.TwoPort;
   Medium.BaseProperties gas;
equation
  InPut.dm   = - OutPut.dm;
  InPut.dmXi = - OutPut.dmXi;
  InPut.dmXi =   InPut.dm * gas.Xi;
  InPut.dH   = - OutPut.dH;
  InPut.dH   = InPut.dm * gas.h;
```

```
end IdealRestriction;
```

### 2.1.5. FuelAirMixture

The FuelAirMixture model is the base class for a set of mixture models. An implementation requires a consistent set of medium models for

- Fuel
- Air
- Unburned gas
- Burned gas

Each implementation is also required to include functions for calculation of burned species. The only thing that has to be replaced when changing gas is the implementation of this model.

```
package FuelAirMixture
  package airMedium extends
    Modelica.Media.IdealGases.MixtureGases.CombustionAir;
  end airMedium;

  package fuelMedium extends
    Modelica.Media.IdealGases.Common.SingleGasNasa(…);
  end fuelMedium;

  package unburnedMedium extends
    Modelica.Media.IdealGases.Common.MixtureGasNasa(…);
  end unburnedMedium;

  package burnedMedium extends
    Modelica.Media.IdealGases.MixtureGases.FlueGasSixComponents(…);
  end burnedMedium;

  function calcBurnedFractions
    input  SI.MassFraction unburned_dmX[unburnedMedium.nX];
    input  SI.Temperature  Tburned;
    output SI.MassFraction burned_dmX[burnedMedium.nX];
  end calcBurnedFractions;
end FuelAirMixture;
```

### 2.1.6. AdiabaticBurner

The AdiabaticBurner model implements heat release by transferring gas from the unburned medium representation to the burned medium. This requires that medium models use `excludeEnthalpyOfFormation = false`. The released energy equals

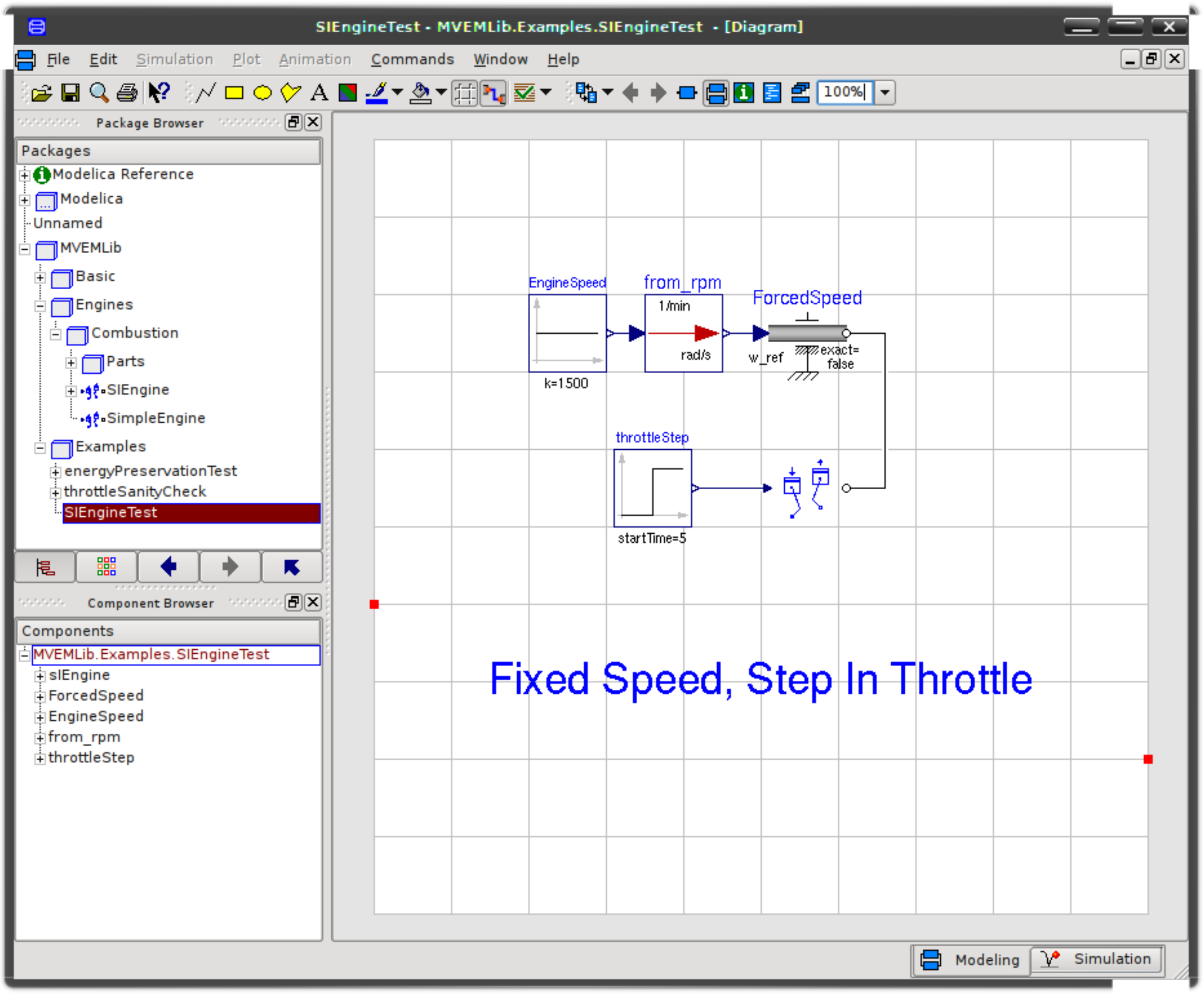
```
dh = burnedMedium.specificInternalEnergy(burnedGas.state)-
     unburnedMedium.specificInternalEnergy(unburnedGas.state);
```

```
model AdiabaticBurner
  replaceable package fuelAirMedium = FuelAirMixture;
  Interfaces.GasPort UnburnedInPut(…);
  Interfaces.GasPort BurnedOutPut(…);
protected
  fuelAirMedium.unburnedMedium.BaseProperties unburnedGas;
  fuelAirMedium.burnedMedium.BaseProperties burnedGas;
equation
  unburnedGas.p  = burnedGas.p;
```

```
   unburnedGas.p  = BurnedOutPut.p;
   unburnedGas.T  = burnedGas.T;
   unburnedGas.T  = BurnedOutPut.T;
   unburnedGas.Xi = UnburnedInPut.dmXi/UnburnedInPut.dm;
   burnedGas.Xi   = BurnedOutPut.dmXi/BurnedOutPut.dm;
   UnburnedInPut.p   = BurnedOutPut.p;
   UnburnedInPut.T   = BurnedOutPut.T;
   UnburnedInPut.Xi  = reference_X[…];//Dummy
   BurnedOutPut.dm   = -UnburnedInPut.dm;
   BurnedOutPut.dH   = -UnburnedInPut.dH;
   BurnedOutPut.dmXi = calcBurnedFractions(UnburnedInPut.dmXi,BurnedOutPut.T);
end AdiabaticBurner;
```

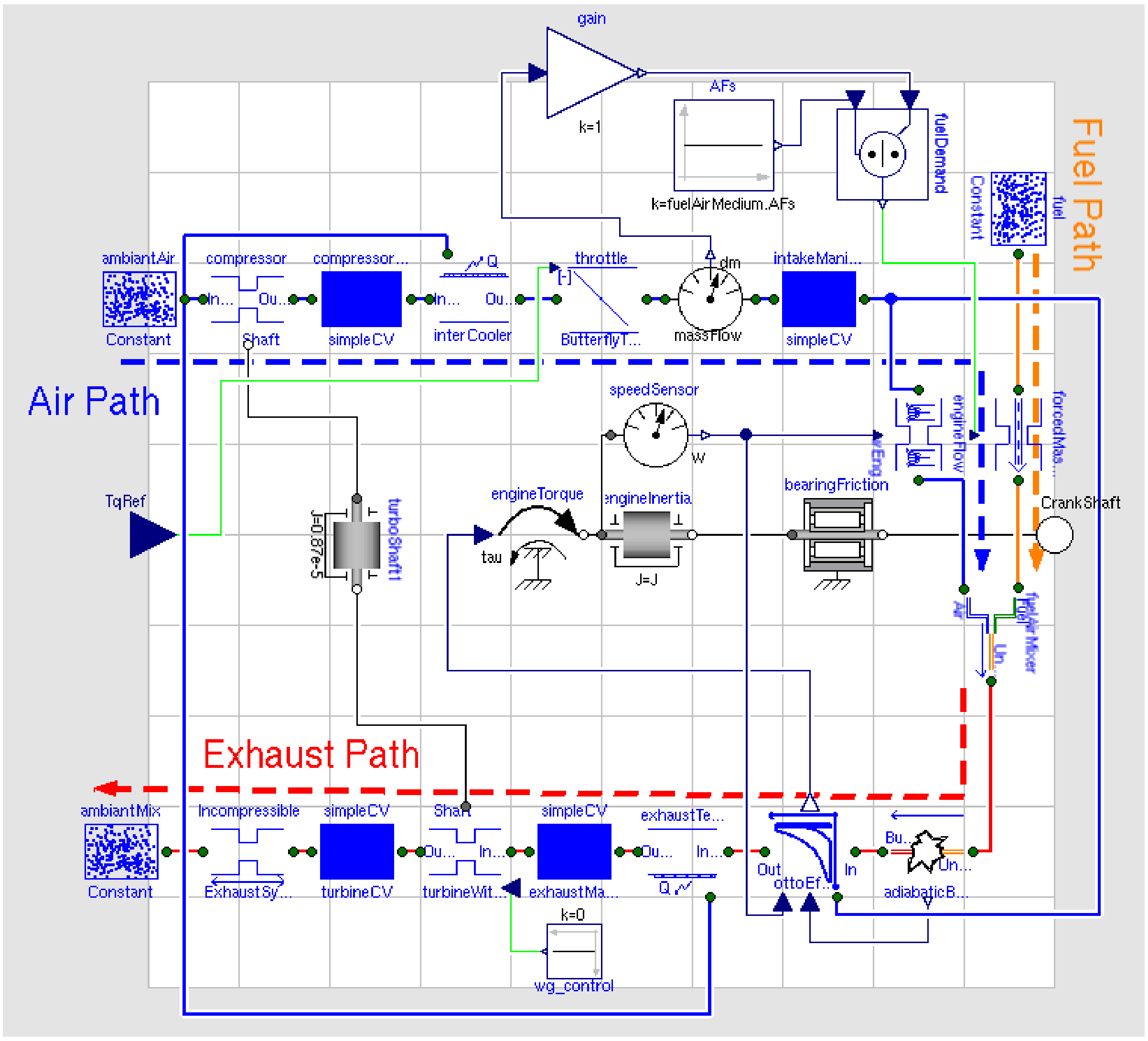
## 3. Simulation Example

As an example of the SIEngine Model is shown below. In this example the throttle position is changed from 20% to 40% while keeping the engine speed at 1500 rpm, a common dynamic test that is performed in an engine test bench.



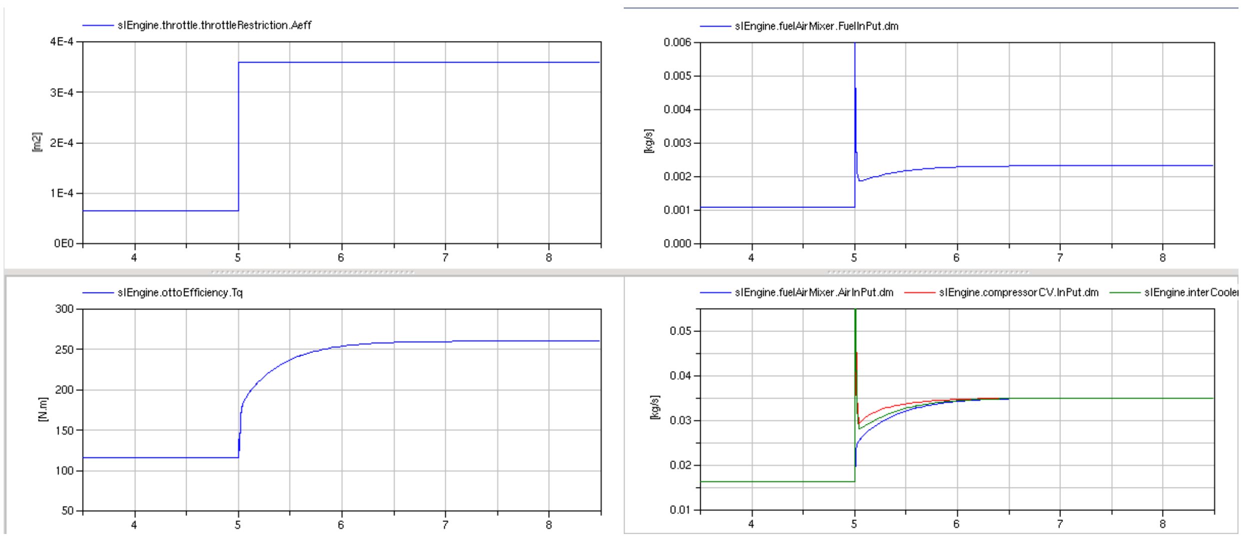
## 3.1. Engine Model

The engine model that is used is a turbocharged 2.3l spark ignited engine as shown in the figure below. Following the air path of the model it has a compressor, intercooler, throttle, intake manifold and a speed-density engine air flow model. All models that have a pressure drop needs to have a control volume or source object upstreams and downstreams. It is thus impossible to connect two throttles in series, while for the intercooler that only represents a temperature drop and the mass flow sensor that just act as a pass through device it is possible to stack them in series together with one pressure drop component. For the fuel flow path a source and an externally controller forced fuel flow is used. In the model the air is represented by Oxygen and Nitrogen while the fuel is a single component gas with ISO-Octane, C8H18. During the mixing they are translated to a unburned representation using a three component gas with Oxygen, Nitrogen, and ISO-Octane. The unburned gas is then combusted which results in a rise in temperature and a translation into a six component gas, Modelica.Media.IdealGases.MixtureGases.FlueGasSixComponents. In the exhaust path a otto-engine efficiency models is used to calculate the torque whereafter the gas exits through a temperature drop pipe and the turbine. As can be seen the Turbine and the Compressor are connected through a turbo-shaft that transfers power from the exhaust gases to the compressor.

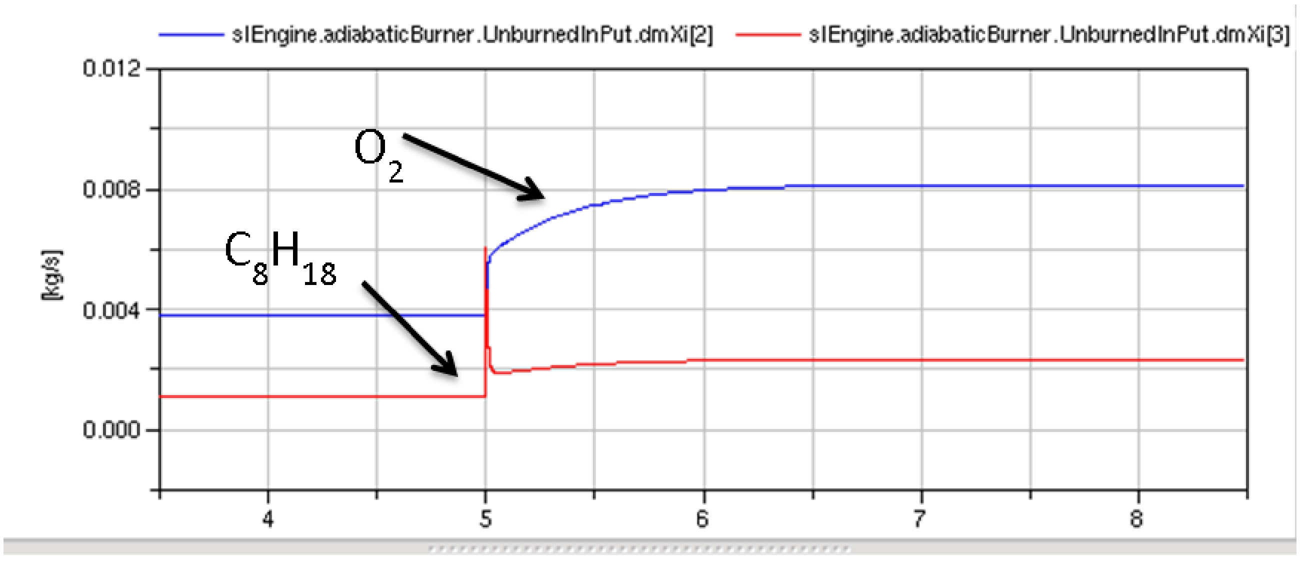


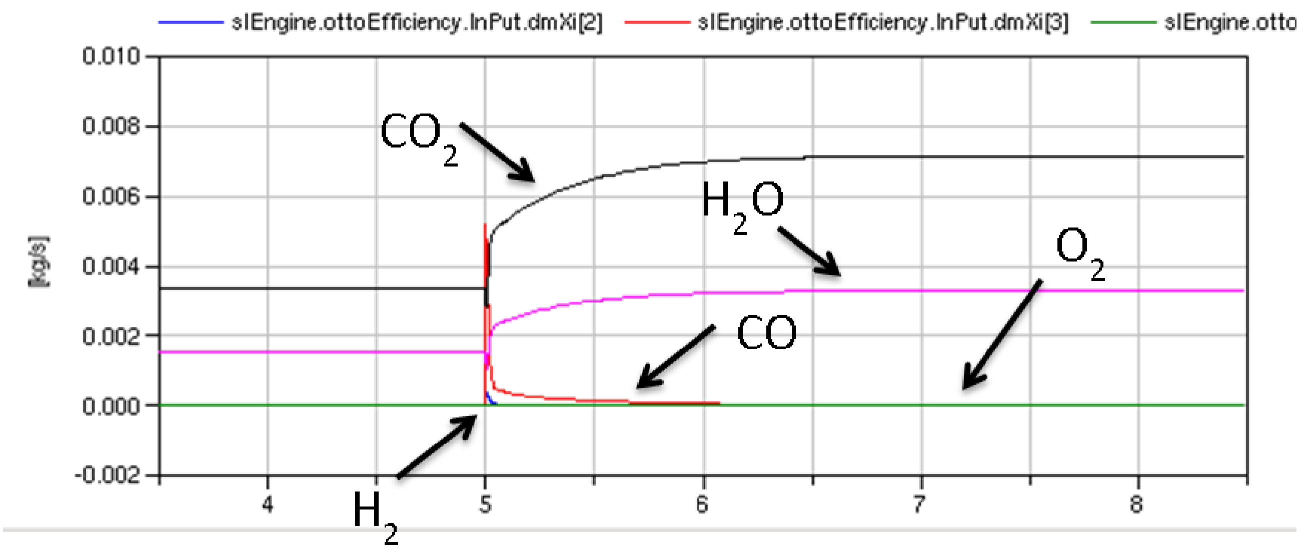
## 3.2. Simulation Results

In the figure below it can be see how a step in throttle area (t.l.) gives increase in engine torque (l.l.). Mass flows for the air path differs between the positions in the engine. The flow to the cylinders does not spike, as in contrast to the compressor flow (l.r.). The spike in compressor flow gives a spike in fuel input from the fuel-controller because it measures air flow at the location close to the compressor instead of the air flow to the actual cylinder.



During the transient in fuel flow the fuel concentration is increased from the stoichiometric value because of the lack of air flow estimation model for the cylinder flow. In the figure below it is shown how the air/fuel mass flows changes during the transient. Because the Nitrogen is assumed to be inert is removed from the graphs for visibility reasons. For the unburned gases a spike in fuel flow can be seen (top) and for the burned gases typical left overs from incomplete combustions such as hydrogen and carbon monoxide increases temporarily.





## 4. Conclusion

- This Library and example shows how the Modelica media models can be used for mean value engine modeling of a SI (and CI) engine.
- The implementation yields a possibility to have different medium models with different complexities in the same framework.
- Only the FuelAirMixture package needs to be replaced to change gas representation.
- During simulation studies different gas models with different complexities can be used at different stages without re-implementing any part of the models. This enables easy reuse of code.
- The use case relies heavily on Media library, specifically replaceable package
- Support in openModelica is as of now limited and the use case thus drives the development by requiring advanced features.